6:20-cv-00260

Exhibit B

6,819,539	T-Mobile's Sale of the Motorola Moto Z2 Force ("The Accused Product")
8. An apparatus comprising: a detection circuit configured to generate a signal having an event condition; and	The accused product utilizes an apparatus comprising: a detection circuit (e.g., a battery monitoring circuit) configured to generate a signal (e.g., voltage or current notification) having an event condition (e.g., if state is high or low). As shown below, the Motorola Moto Z2 Force utilizes a Qualcomm Snapdragon 835 processor. 

EXHIBIT B

Specs

- **Battery**
  - Usage time: 10 hours
  - Standby time: 18 hours
  - Battery size/type: 2730 mAh
  - Fast charging type: 15W TurboPower
- **Keyboard**
  - Touch screen with on-screen keyboard
- **Memory**
  - 4 GB RAM, 64 GB ROM
  - Supports 256 MB up to 2 TB MicroSD card
- **Operating System**
  - Android
- **Processor**
  - Qualcomm® Snapdragon™ 835 Octa-Core, MSM 8998
- **Anti-theft**
  - Yes
- **Advanced messaging**
  - Yes
- **Device Unlock App**
  - Yes
- **Emergency Alerts (WEA)**
  - See T-Mobile.com/WEA
- **SIM card**
  - Nano-SIM
- **System Manager (Carrier IQ)**
  - Yes
- **T-Mobile Video Calling**
  - No - uses Google Duo

https://www.t-mobile.com/support/devices/android/motorola-moto-z2-force-edition/tech-specs-motorola-moto-z2-force-edition

As shown below, the Snapdragon 835 includes a battery monitoring circuit that generates a signal based upon the occurrence of a certain condition (in this case voltage variances for normal values).

https://www.qualcomm.com/products/snapdragon-835-mobile-platform

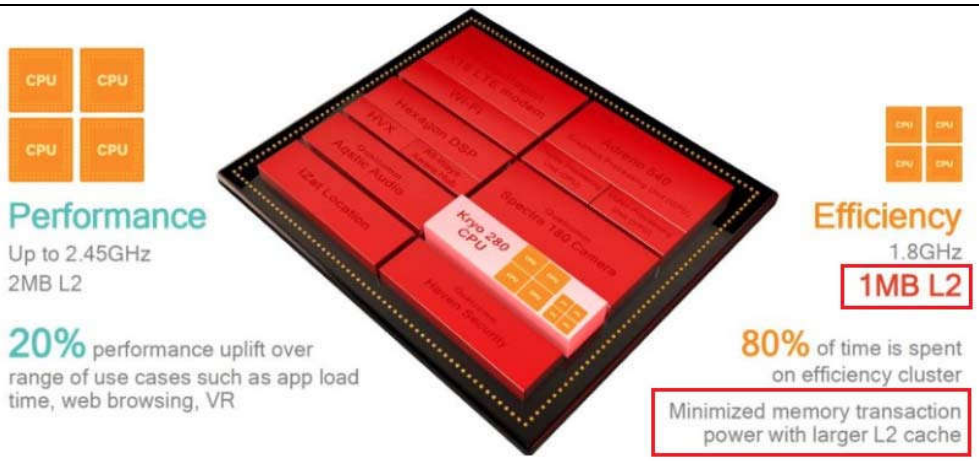
	Snapdragon 835 mobile platform advancements: + Snapdragon X16 LTE modem: mobile connectivity with LTE download speeds up to 1 Gbps, multi-gigabit 802.11ad, and integrated 2x2 802.11ac Wi-Fi with MU-MIMO + Qualcomm® Quick Charge™ 4 technology: 20% faster, 30% more efficient than our previous generation, charge from zero to up to 50% in 15 minutes[2] + Qualcomm® Adreno™ 540 GPU with visual processing subsystem: Advanced 3-D graphics rendering and up to 60X more colors help deliver life-like visuals for immersive experiences[1] + Qualcomm Spectra™ 180 Camera ISP: Dual 14-bit ISPs support up to 32MP single or dual 16MP cameras for the ultimate photography and videography experience + Qualcomm® Hexagon™ 682 DSP: Support for latest Machine Learning frameworks and image processing. Includes Hexagon Vector eXtensions and Qualcomm All-Ways Aware™ technology utilizing connectivity and sensors https://www.qualcomm.com/media/documents/files/snapdragon-835-mobile-platform-product-brief.pdf

```
5006.        qcom,bcl {
5007.            compatible = "qcom,bcl";
5008.            qcom,bcl-enable;
5009.            qcom,bcl-framework-interface;
5010.            qcom,bcl-freq-control-list = <0x1a 0x1b 0x1c 0x1d>;
5011.            qcom,bcl-hotplug-list = <0x1a 0x1b 0x1c 0x1d>;
5012.            qcom,bcl-soc-hotplug-list = <0x1a 0x1b 0x1c 0x1d>;
5013.
5014.            qcom,ibat-monitor {
5015.                qcom,low-threshold-uamp = <0x33e140>;
5016.                qcom,high-threshold-uamp = <0x401640>;
5017.                qcom,mitigation-freq-khz = <0x8ca00>;
5018.                qcom,vph-high-threshold-uv = <0x3567e0>;
5019.                qcom,vph-low-threshold-uv = <0x325aa0>;
5020.                qcom,soc-low-threshold = <0xa>;
5021.                qcom,thermal-handle = <0xa0>;
5022.            };
5023.        };
```

https://pastebin.com/U0i7nP4P

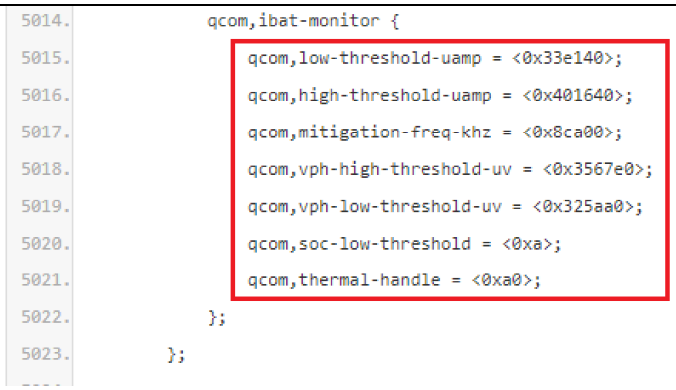
<table>
<tr>
<td></td>
<td>
<pre>
564 bcl->btm_vph_adc_param.btm_ctx = bcl;
565 bcl->btm_vph_adc_param.threshold_notification = bcl_vph_notification;
566 bcl->btm_vph_adc_param.channel = bcl->btm_vph_chan;
</pre>
Reports a volatage value
<pre>
1381 bcl->btm_ibat_adc_param.btm_ctx = bcl;
1382 bcl->btm_ibat_adc_param.threshold_notification = bcl_ibat_notification;
1383 bcl->btm_ibat_adc_param.channel = bcl->btm_ibat_chan;
</pre>
Reports a current value
<pre>
536 static void bcl_ibat_notification(enum qpnp_tm_state state, void *ctx);
537 static void bcl_vph_notification(enum qpnp_tm_state state, void *ctx);
</pre>
https://android.googlesource.com/kernel/msm/+/refs/heads/android-msm-angler-3.10-nougat/drivers/power/battery_current_limit.c
<pre>
707 enum qpnp_tm_state {
708   ADC_TM_HIGH_STATE = 0,
709   ADC_TM_COOL_STATE = ADC_TM_HIGH_STATE,
710   ADC_TM_LOW_STATE,
711   ADC_TM_WARM_STATE = ADC_TM_LOW_STATE,
712   ADC_TM_STATE_NUM,
713 };
</pre>
https://android.googlesource.com/kernel/msm/+/refs/heads/android-msm-asus-3.10-nougat-mr1-wear-release/include/linux/qpnp/qpnp-adc.h
</td>
</tr>
<tr>
<td>a storage circuit configured to store said event;</td>
<td>The accused product comprises a storage circuit (e.g., L2 cache) configured to store said event (e.g., if state is high or low).

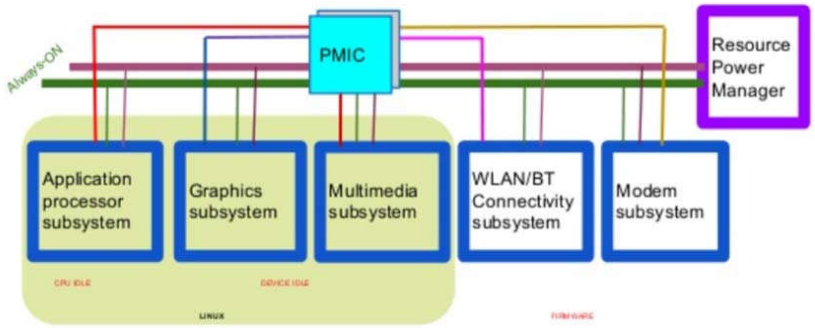
As shown below, the Snapdragon 835 includes an L2 cache that stores voltage variance events.</td>
</tr>
</table>

	 https://www.androidauthority.com/qualcomm-details-snapdragon-835-735688/
a table configured to store a plurality of event types; and	The accused product comprises a table (e.g., a table containing various thresholds) configured to store a plurality of event types (e.g., if state is high or low). As shown in the code below, the Snapdragon 835 utilizes a table that defines various voltage conditions and their table corresponding thresholds.

```
5006.        qcom,bcl {
5007.            compatible = "qcom,bcl";
5008.            qcom,bcl-enable;
5009.            qcom,bcl-framework-interface;
5010.            qcom,bcl-freq-control-list = <0x1a 0x1b 0x1c 0x1d>;
5011.            qcom,bcl-hotplug-list = <0x1a 0x1b 0x1c 0x1d>;
5012.            qcom,bcl-soc-hotplug-list = <0x1a 0x1b 0x1c 0x1d>;
5013.
5014.            qcom,ibat-monitor {
5015.                qcom,low-threshold-uamp = <0x33e140>;
5016.                qcom,high-threshold-uamp = <0x401640>;
5017.                qcom,mitigation-freq-khz = <0x8ca00>;
5018.                qcom,vph-high-threshold-uv = <0x3567e0>;
5019.                qcom,vph-low-threshold-uv = <0x325aa0>;
5020.                qcom,soc-low-threshold = <0xa>;
5021.                qcom,thermal-handle = <0xa0>;
5022.            };
5023.        };
```

https://pastebin.com/U0i7nP4P

	 https://pastebin.com/U0i7nP4P
a circuit configured to (i) reset when said event condition is a first predetermined type and (ii) implement recover action when said event condition is a second predetermined type, wherein said first and second predetermined types are determined in response to a comparison of said event to said plurality of event types stored in said table.	The accused product comprises a circuit (e.g., resource power manager circuit) configured to (i) reset (e.g., cpu_down) when said event condition is a first predetermined type (e.g., when bcl_soc_state == BCL_LOW_THRESHOLD OR bcl_vph_state == BCL_LOW_THRESHOLD) and (ii) implement recover action (e.g., cpu_up) when said event condition is a second predetermined type (e.g., when bcl_soc_state is not equal to BCL_LOW_THRESHOLD, bcl_vph_state is not equal to BCL_LOW_THRESHOLD and bcl_ibat_state is not equal to BCL_HIGH_THRESHOLD), wherein said first and second predetermined types are determined in response to a comparison of said event to said plurality of event types stored in said table (e.g. the comparison of collected values with stored thresholds).

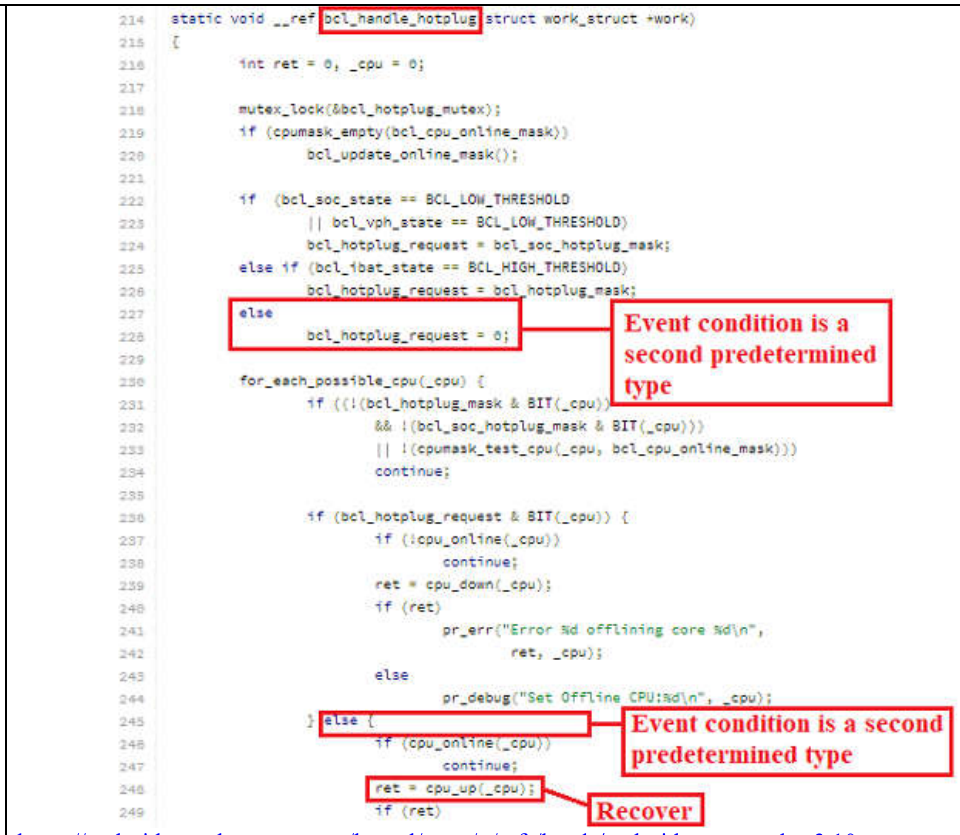
	4 Resource Power Manager (RPM) 5 6 RPM is a dedicated hardware engine for managing shared SoC resources, 7 which includes buses, clocks, power rails, etc. The goal of RPM is 8 to achieve the maximum power savings while satisfying the SoC's 9 operational and performance requirements. RPM accepts resource 10 requests from multiple RPM masters. It arbitrates and aggregates the 11 requests, and configures the shared resources. The RPM masters are 12 the application processor, the modem processor, as well as some 13 hardware accelerators. https://android.googlesource.com/kernel/msm/+/android-7.1.0_r0.2/Documentation/arm/msm/rpm.txt  https://www.slideshare.net/linaroorg/lcu14-210-qualcomm-snapdragon-power-management-unique-challenges-for-power-frameworks

```
#ifdef CONFIG_SMP
static void __ref bcl_handle_hotplug(struct work_struct *work)
{
        int ret = 0, _cpu = 0;

        mutex_lock(&bcl_hotplug_mutex);
        if (cpumask_empty(bcl_cpu_online_mask))
                bcl_update_online_mask();

        if  (bcl_soc_state == BCL_LOW_THRESHOLD
                || bcl_vph_state == BCL_LOW_THRESHOLD)
                bcl_hotplug_request = bcl_soc_hotplug_mask;
        else if (bcl_ibat_state == BCL_HIGH_THRESHOLD)
                bcl_hotplug_request = bcl_hotplug_mask;
        else
                bcl_hotplug_request = 0;

        for_each_possible_cpu(_cpu) {
                if ((!(bcl_hotplug_mask & BIT(_cpu))
                        && !(bcl_soc_hotplug_mask & BIT(_cpu)))
                        || !(cpumask_test_cpu(_cpu, bcl_cpu_online_mask)))
                        continue;

                if (bcl_hotplug_request & BIT(_cpu)) {
                        if (!cpu_online(_cpu))
                                continue;
                        ret = cpu_down(_cpu);
                        if (ret)
```

Event condition is a first predetermined type

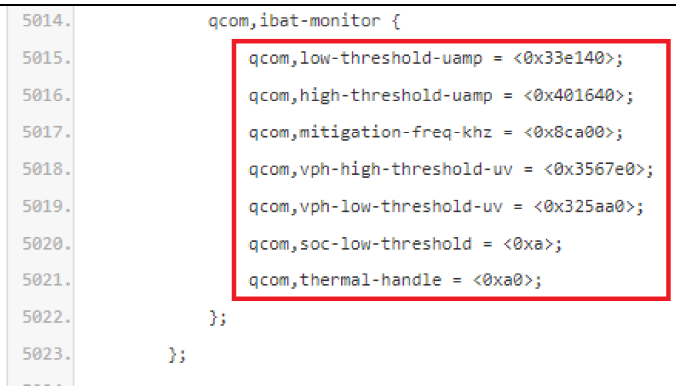
Reset

https://android.googlesource.com/kernel/msm/+/refs/heads/android-msm-angler-3.10-nougat/drivers/power/battery_current_limit.c




```
214 static void __ref bcl_handle_hotplug(struct work_struct *work)
215 {
216         int ret = 0, _cpu = 0;
217
218         mutex_lock(&bcl_hotplug_mutex);
219         if (cpumask_empty(bcl_cpu_online_mask))
220                 bcl_update_online_mask();
221
222         if  (bcl_soc_state == BCL_LOW_THRESHOLD
223                 || bcl_vph_state == BCL_LOW_THRESHOLD)
224                 bcl_hotplug_request = bcl_soc_hotplug_mask;
225         else if (bcl_ibat_state == BCL_HIGH_THRESHOLD)
226                 bcl_hotplug_request = bcl_hotplug_mask;
227         else
228                 bcl_hotplug_request = 0;
229
230         for_each_possible_cpu(_cpu) {
231                 if ((!(bcl_hotplug_mask & BIT(_cpu)))
232                         && !(bcl_soc_hotplug_mask & BIT(_cpu)))
233                         || !(cpumask_test_cpu(_cpu, bcl_cpu_online_mask)))
234                         continue;
235
236                 if (bcl_hotplug_request & BIT(_cpu)) {
237                         if (!cpu_online(_cpu))
238                                 continue;
239                         ret = cpu_down(_cpu);
240                         if (ret)
241                                 pr_err("Error %d offlining core %d\n",
242                                         ret, _cpu);
243                         else
244                                 pr_debug("Set Offline CPU:%d\n", _cpu);
245                 } else {
246                         if (cpu_online(_cpu))
247                                 continue;
248                         ret = cpu_up(_cpu);
249                         if (ret)
```

https://android.googlesource.com/kernel/msm/+/refs/heads/android-msm-angler-3.10-nougat/drivers/power/battery_current_limit.c

	 https://pastebin.com/U0i7nP4P Threshold Values from the table (dtsi) are imported into the battery_current_limit module thru a record data type (bcl).

```
5014.            qcom,ibat-monitor {
5015.                qcom,low-threshold-uamp = <0x33e140>;
5016.                qcom,high-threshold-uamp = <0x401640>;
5017.                qcom,mitigation-freq-khz = <0x8ca00>;
5018.                qcom,vph-high-threshold-uv = <0x3567e0>;
5019.                qcom,vph-low-threshold-uv = <0x325aa0>;
5020.                qcom,soc-low-threshold = <0xa>;
5021.                qcom,thermal-handle = <0xa0>;
5022.            };
5023.        };
```

```
1519
1520        BCL_FETCH_DT_U32(ibat_node, key, "qcom,low-threshold-uamp", ret,
1521                bcl->ibat_low_thresh.trip_value, ibat_probe_exit);
1522        BCL_FETCH_DT_U32(ibat_node, key, "qcom,high-threshold-uamp", ret,
1523                bcl->ibat_high_thresh.trip_value, ibat_probe_exit);
1524        BCL_FETCH_DT_U32(ibat_node, key, "qcom,mitigation-freq-khz", ret,
1525                bcl->bcl_p_freq_max, ibat_probe_exit);
1526        BCL_FETCH_DT_U32(ibat_node, key, "qcom,vph-high-threshold-uv", ret,
1527                bcl->vbat_high_thresh.trip_value, ibat_probe_exit);
1528        BCL_FETCH_DT_U32(ibat_node, key, "qcom,vph-low-threshold-uv", ret,
1529                bcl->vbat_low_thresh.trip_value, ibat_probe_exit);
1530        BCL_FETCH_DT_U32(ibat_node, key, "qcom,soc-low-threshold", ret,
1531                soc_low_threshold, ibat_probe_exit);
```

The values of the table are now inside the record, bcl. The State of Charge low threshold is saved in a variable soc_low_threshold.

```
174         /* BCL Peripheral monitor parameters */
175         struct bcl_threshold ibat_high_thresh;
176         struct bcl_threshold ibat_low_thresh;
177         struct bcl_threshold vbat_high_thresh;
178         struct bcl_threshold vbat_low_thresh;
179         uint32_t bcl_p_freq_max;
180 };
```

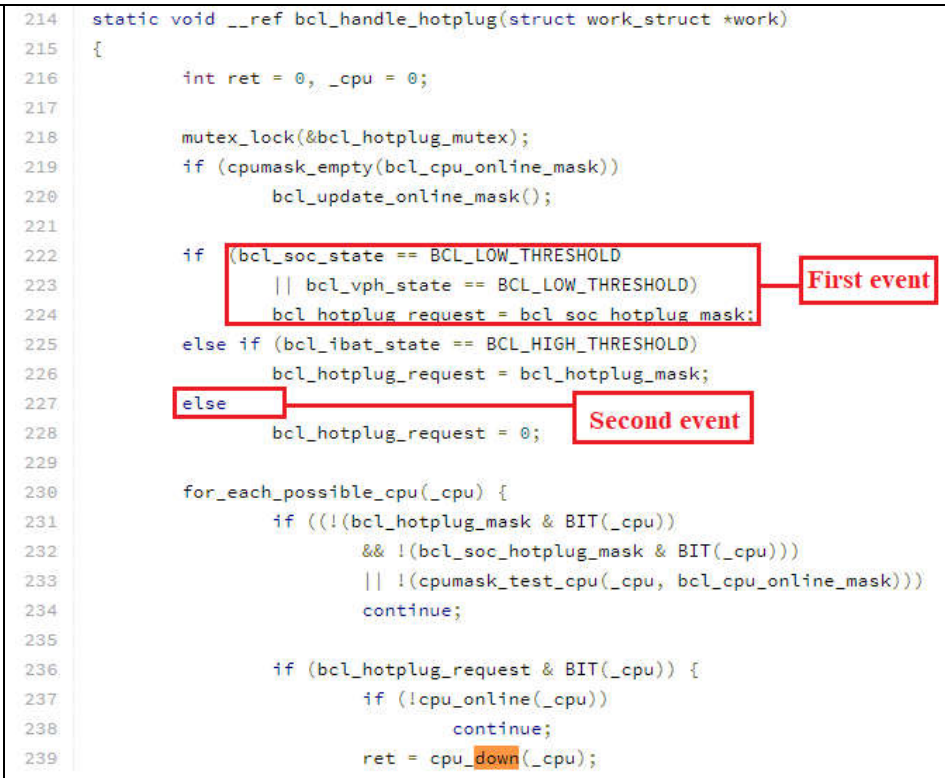
Different possible event types

https://android.googlesource.com/kernel/msm/+/refs/heads/android-msm-angler-3.10-nougat/drivers/power/battery_current_limit.c

```
17  #define BCL_NAME_MAX_LEN 20
18
19  enum bcl_trip_type {
20          BCL_HIGH_TRIP,
21          BCL_LOW_TRIP,
22          BCL_TRIP_MAX,
23  };
```

https://android.googlesource.com/kernel/msm/+/refs/heads/android-msm-angler-3.10-nougat/include/linux/msm_bcl.h

```
31  struct bcl_threshold {
32          int                     trip_value;
33          enum bcl_trip_type      type;
34          void                    *trip_data;
35          void (*trip_notify)     (enum bcl_trip_type, int, void *);
36  };
```




```
214  static void __ref bcl_handle_hotplug(struct work_struct *work)
215  {
216          int ret = 0, _cpu = 0;
217
218          mutex_lock(&bcl_hotplug_mutex);
219          if (cpumask_empty(bcl_cpu_online_mask))
220                  bcl_update_online_mask();
221
222          if (bcl_soc_state == BCL_LOW_THRESHOLD
223              || bcl_vph_state == BCL_LOW_THRESHOLD)
224              bcl_hotplug_request = bcl_soc_hotplug_mask;
225          else if (bcl_ibat_state == BCL_HIGH_THRESHOLD)
226                  bcl_hotplug_request = bcl_hotplug_mask;
227          else
228                  bcl_hotplug_request = 0;
229
230          for_each_possible_cpu(_cpu) {
231                  if ((!(bcl_hotplug_mask & BIT(_cpu))
232                          && !(bcl_soc_hotplug_mask & BIT(_cpu)))
233                          || !(cpumask_test_cpu(_cpu, bcl_cpu_online_mask)))
234                          continue;
235
236                  if (bcl_hotplug_request & BIT(_cpu)) {
237                          if (!cpu_online(_cpu))
238                                  continue;
239                          ret = cpu_down(_cpu);
```

https://android.googlesource.com/kernel/msm/+/refs/heads/android-msm-angler-3.10-nougat/drivers/power/battery_current_limit.c

The new values of bcl_vph_state and bcl_ibat_state are compared against the threshold values from the table.